Primitive vs. Reference Types

- We’ve seen Java’s 4 primitive types: int, double, boolean, char
- Types other than the primitive types are known as reference types
  - Used for objects
- Examples of reference variables:
  - Dog d1; Counter c1; Player mario; String name;
  - Note: String is an object not primitive type

Memory: Stack and Heap

- When we run Java programs, memory is allocated for variables and objects
- Understanding how this memory is managed helps us understand how Java works
- The JVM uses two kinds of memory: stack and heap
- The stack is used to store variables of primitive type:
  - When created in the DrJava interactions pane
  - During method calls
- The heap is used to store objects

How the Stack Works

<table>
<thead>
<tr>
<th>DrJava Interactions</th>
<th>Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt; int x;</td>
<td><img src="image" alt="Stack Diagram" /></td>
</tr>
<tr>
<td>&gt; x = 5;</td>
<td><img src="image" alt="Stack Diagram" /></td>
</tr>
<tr>
<td>&gt; double min = 0.5;</td>
<td><img src="image" alt="Stack Diagram" /></td>
</tr>
<tr>
<td>&gt; boolean done = false;</td>
<td><img src="image" alt="Stack Diagram" /></td>
</tr>
</tbody>
</table>

Note: Variables are added in the order they are declared
### Reference Type
- In Java, no variable can ever hold an entire object.
- One variable can only contain one thing.
- Object consists of multiple of data/state and hence stored on **heap**.
- The term *reference* is used because it refers to a memory location where the object lives.
- The variable of reference type is used to access the object.
- The value of reference variable is either “null” or a “heap address.”
  - *null* means currently not pointing at any location.

### Value of a Reference Variable
Example:
```java
> Counter c1;
> c1
null
> c1 = new Counter();
> c1
Counter@e05ad6
```
- `e05ad6` is location in memory where `c1` resides.
  - `e05ad6` hexadecimal (base 16) number.
  - This location will different on your computer.
- We don’t have to (and can’t) deal with these hex numbers directly.
  - Convenience of using variables.

### How the Heap Works

<table>
<thead>
<tr>
<th>DrJava Interactions</th>
<th>Stack and Heap</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>int x = 99;</code></td>
<td><img src="null" alt="null" /></td>
</tr>
<tr>
<td><code>Counter c1;</code></td>
<td><code>c1</code></td>
</tr>
<tr>
<td><code>null</code></td>
<td><code>null</code></td>
</tr>
<tr>
<td><code>c1 = new Counter();</code></td>
<td><code>c1</code></td>
</tr>
<tr>
<td><code>Counter@2f996f</code></td>
<td></td>
</tr>
<tr>
<td><code>c1.incrementCount();</code></td>
<td><code>c1</code></td>
</tr>
<tr>
<td><code>Dog d1 = new Dog(&quot;Lassie&quot;, 5);</code></td>
<td><code>d1</code></td>
</tr>
<tr>
<td><code>Dog@83d8be</code></td>
<td></td>
</tr>
<tr>
<td><code>d1.getAge()</code></td>
<td>5</td>
</tr>
</tbody>
</table>

### Aliases
- Two or more references can point to the same object.
- These references are then known as aliases.
- Example (In DrJava Interactions Pane):  
  ```java
  > Dog d1 = new Dog("Lassie", 5);
  > d1
  Dog@83d8be
  > Dog d2 = d1;
  > d2
  Dog@83d8be
  > d1.getAge()
  5
  > d2.getAge()
  5
  ```
String

- A sequence of characters
- A String is a built-in Java object type
- Java provides this type because it’s used so frequently
- Examples of String creation:

```java
> String s1 = new String("hello");
> String s2 = "hello"; // commonly used shortcut
> s2 + " you!"
"hello you!"
> s2 = "The result is " + 100;
> System.out.println(s2);
"The result is 100"
```

String (contd..)

- We do not get heap address of String reference
- Later, when we learn “Inheritance” it will be clear

DrJava Interactions

<table>
<thead>
<tr>
<th>String c1 = new String(&quot;Hill&quot;);</th>
</tr>
</thead>
</table>

Stack and Heap

```java
> String c1 = new String("Hill");
> c1
"Hill"
```

Note:

- We do not get heap address of String reference
- Later, when we learn “Inheritance” it will be clear